

A Systematic Study of Compression Ordering for Large Language Models

Shivansh Chhawri^a, Rahul Mahadik^a, Suparna Rooj^a

^a*equal contribution*

Abstract

Large Language Models (LLMs) require substantial computational resources, making model compression essential for efficient deployment in constrained environments. Among the dominant compression techniques—knowledge distillation, structured pruning, and low-bit quantization—their individual effects are well studied, but their interactions and optimal sequencing remain unclear. This work systematically examines how these techniques perform both independently and in combination when applied to the Qwen2.5-3B model. We evaluate multiple compression pipelines, including single, and proposed three-technique sequences, using perplexity, G-Eval, clarity, prompt alignment, and compression ratio as metrics. Our experiments show that quantization provides the greatest standalone compression, while pruning introduces moderate quality degradation. Critically, the ordering of techniques significantly affects the final model quality: the sequence Pruning → Knowledge Distillation → Quantization (P-KD-Q) yields the best balance, achieving a 3.68× compression ratio while preserving strong instruction-following and language understanding capabilities. Conversely, pipelines applying quantization early suffer severe performance degradation due to irreversible information loss that impairs subsequent training. Overall, this study offers practical insight into designing effective, ordering-aware compression pipelines for deploying LLMs in resource-limited settings.

Keywords: Knowledge Distillation, large language models, pruning, quantization, qwen

1. Introduction

Over the past decade, artificial intelligence has experienced an extraordinary surge in capability and adoption, evolving from traditional machine

learning methods to powerful deep learning systems and, more recently, to advanced generative models. This rapid growth has been fueled by breakthroughs in neural network architectures, large-scale computing, and the availability of massive datasets. Today, AI is profoundly transforming diverse domains—including healthcare [1], finance [2], manufacturing [3], defense, etc. Despite remarkable progress in AI, machines do not inherently understand or generate human language without sophisticated computational models. Language modeling [4], a core task in natural language processing, focuses on predicting the next token—word or character—within a sequence, enabling systems to capture linguistic structure and contextual meaning which before the emergence of AI was a critical bottleneck [5]. Large language models (LLMs) have rapidly grown in popularity due to their ability to perform a wide range of tasks such as text generation, reasoning, summarization after the publication of the attention based transformer model [6] research paper, which is the base of all the LLMs. The emergence of LLMs, image generation systems, and diffusion-based techniques highlights the accelerating pace of innovation, marking a significant shift in the way intelligent systems are developed and deployed.

Research on large language models has advanced rapidly since the introduction of transformer-based architectures, beginning with OpenAI’s Generative Pre-trained Transformer (GPT) series, where GPT [7], GPT-2 [8] and GPT-3 [9] demonstrated that scaling the size of the model and training data significantly improve language understanding and generation. This paradigm was further expanded by Google’s BERT [10] and T5 [11], which highlighted the effectiveness of bidirectional and sequence-to-sequence pre-training strategies. Meta’s LLaMA models [12] introduced a family of efficient high-performance LLMs trained on carefully curated datasets, enabling strong results with fewer parameters. Concurrently, Anthropic’s Claude models [13] emphasized constitutional AI and safer alignment techniques for large-scale language systems. More recently, models such as PaLM [14] and Qwen [15] have continued to push the boundaries of multilingual reasoning, instruction following, and generalization. Together, these works form a rich body of literature that charts the evolution of LLMs from early transformer models to today’s powerful foundation models used across research and industry.

Although large language models have demonstrated remarkable performance across diverse tasks, their increasing scale brings significant challenges that limit their practical deployment. Modern LLMs require substantial computational resources, large memory footprints, and high energy consumption,

making them difficult to run on edge devices, real-time systems, or cost-constrained environments. Their inference latency can hinder interactive applications, and frequent retraining or fine-tuning exacerbates the computational burden. These limitations have created a strong need for efficient model compression techniques, such as pruning [16], knowledge distillation [17], and quantization [18], which aim to reduce model size, improve inference speed, and lower resource requirements while maintaining accuracy. Motivated by these issues, this research explores how such optimization strategies can make large language models more scalable, accessible, and practical for real-world use. Quantization in deep learning is an optimization technique that reduces the precision of a model’s numerical values, typically converting them from floating-point numbers to lower-precision integers. The authors in [18, 19] have published a comprehensive study on the quantization of LLMs range mapping viz, affine quantization, scale quantization, quantization techniques viz., post-training quantization, quantization-aware training, weight quantization and activation-aware weight quantization. Although post-training quantization techniques improve LLM’s computational efficiency and memory footprint, their hand-crafted quantization settings result in poor performance, particularly in very low-bit quantization. This issue is addressed in omnidirectionally calibrated quantization i.e., OmniQuant technique for LLMs. To address the issue of substantial training resources in quantization aware training, effientQAT method was proposed consisting of Block-wise training of all parameters and end-to-end training of quantization parameters [20]. Authors in [21] applied low rank adaptation along with quantization aware training simultaneously which reduces the difference between the full-precision and quantized models and greatly enhances generalization in subsequent challenges. Pruning in deep learning is a technique to reduce the size and complexity of a neural network by removing less important parameters, like weights, neurons, or entire layers. In one approach proposed, each weight matrix is parameterized using its low-rank factorization and adaptively rank-1 components are eliminated during training [22]. A novel and effective pruning method titled Wanda (pruning by Weights and activations), designed to induce sparsity in pretrained LLMs [16]. A batched greedy pruning method names SlimGPT for rapid and near-optimal pruning is proposed which enhances the accuracy of head-wise pruning error estimation through grouped Cholesky decomposition [23]. Authors in [24] proposed Fluctuation-based Adaptive Structured Pruning which formulated structured importance metrics, adaptively searched the global compressed

model, and implemented compensation mechanisms to mitigate performance loss.

While individual compression techniques—quantization [25], pruning [26], and knowledge distillation [27], have been extensively studied in isolation, real-world deployment scenarios often require combining multiple techniques to achieve aggressive compression ratios while maintaining acceptable performance [28]. However, the existing literature provides limited guidance on the optimal ordering of these techniques when applied sequentially on small scale LLMs. Recent surveys [29] identify this as a critical gap: different orderings may exhibit synergistic or antagonistic interactions, and certain sequences may be infeasible due to technical constraints (e.g., quantization’s incompatibility with gradient-based training). The foundational "Deep Compression" work in [30] used a pipeline of pruning, quantization, and encoding to drastically reduce the AlexNet model’s size. This same principle of applying techniques in a sequence can be extended to modern small-scale LLMs. This work systematically explores compression technique orderings to identify optimal strategies for practitioners deploying compressed LLMs in resource-constrained environments.

This research paper is organized as follows: Section 2 describes the methodology explaining single strategy baseline compression techniques in section 2.1. This includes knowledge distillation in section 2.1.1, pruning in section 2.1.2, and quantization in section 2.1.3. We propose our compression ordering strategy in section 2.2 with specific focus on 6 three-technique sequences explained in section 2.2.1. Section 3 discussed the results and the analysis part and section 4 concludes the paper.

2. Methodology

The methodology begins by outlining each individual compression strategy—knowledge distillation, pruning and quantization—highlighting how each technique independently reduces model size or computation while maintaining acceptable performance. However, relying on any single method alone introduces limitations, such as accuracy degradation from aggressive quantization, reduced representational capacity in distilled models, or structural instability caused by extensive pruning. To address these shortcomings, we propose a sequential three-stage pipeline that integrates the strengths of all three approaches. We leverage multiple combinations of knowledge distillation, pruning in various orders.

2.1. Compression Techniques

2.1.1. Knowledge Distillation (KD)

Knowledge distillation is an effective model compression technique used to transfer the capabilities of a large, high-performing language model (teacher) into a smaller, more efficient model (student). In the context of LLMs, this process involves guiding the student model to mimic the teacher’s behavior by learning from its soft predictions, hidden representations, or intermediate reasoning patterns. Through this targeted supervision, the student model can capture much of the teacher’s linguistic and reasoning ability while requiring significantly fewer parameters and computational resources.

In this paper, knowledge distillation was performed using Qwen2.5-7B [31] as the teacher model to guide the training of compressed student models, following the framework established in [27]. The teacher model was loaded in 8-bit precision using BitsAndBytes quantization [32] to manage GPU memory constraints while maintaining high-quality probability distributions.

Loss Function: The distillation loss combined two components following the dual-objective approach of [27]:

1. Task Loss: Standard cross-entropy loss between student predictions and ground-truth labels
2. Distillation Loss: KL divergence between teacher and student probability distributions.

The combined loss was formulated as:

$$L_{\text{total}} = (1 - \alpha)L_{\text{task}} + \alpha L_{\text{distill}} \quad (1)$$

where $\alpha = 0.3$, giving 70% weight to task performance and 30% to teacher mimicry. This weighting scheme balances learning from both hard labels and soft targets, as recommended in the distillation literature [33, 34]

Temperature Scaling: Both teacher and student logits were scaled by temperature $T = 4.0$ before computing softmax distributions, following [27]. Temperature scaling smooths the probability distributions and reveals more information about the teacher’s learned similarities between tokens [35]. The distillation loss was scaled by $T^2 = 16$ to compensate for the magnitude reduction caused by temperature scaling.

Masking and Vocabulary Handling: Attention masks were applied during KL divergence computation to ensure that padding tokens did not contribute to the loss, following standard practices in transformer training

[36]. When teacher and student models had mismatched vocabulary sizes, logits were truncated to the minimum shared vocabulary size to enable proper distribution comparison.

2.1.2. Structured Pruning (P)

Pruning reduces the size and computational cost of LLMs by removing weights or neurons that contribute minimally to the model’s predictions. By eliminating these less impactful parameters, pruning helps streamline the network, leading to faster inference and lower memory requirements while aiming to preserve overall performance. Structural pruning goes a step further by removing entire components—such as attention heads, neurons, or even blocks—rather than individual weights.

Structured pruning targeted the feed-forward network layers within each transformer block, specifically the intermediate dimension of the gate_proj, up_proj, and down_proj layers, following the approach referenced in [37] and recently adapted for LLMs in [26]. A pruning ratio of 30% was applied uniformly across all layers, consistent with ratios shown to maintain model quality in recent pruning studies [16, 38].

Importance Score Calculation: Neuron importance was determined using a hybrid metric combining weight magnitude and activation statistics, inspired by the first-order Taylor expansion approach of [39] and adapted for transformer architectures:

- Weight Magnitude: Sum of absolute weights for each neuron [40].

$$I_{weight} = |W_{gate}[i, :]| + |W_{up}[i, :]| \quad (2)$$

- Activation Magnitude: Average activation across calibration samples. $I_{activation} = E[|h_i|]$ over calibration batches
- Combined Score: $I_{combined} = 0.5 \times I_{weightnorm} + 0.5 \times I_{activationnorm}$ where normalization was performed by dividing by the maximum value within each layer to ensure equal scaling across different network depths [26].

For each layer, neurons were ranked by combined importance score, and the bottom 30% were removed. Structured pruning was performed by selecting top 70% neurons (keep_indices) based on importance, and reconstructing gate_proj and up_proj with reduced output dimensions, down_proj with reduced input dimension to maintain dimensionality consistency.

2.1.3. Quantization (Q)

Quantization reduces the memory footprint and computational load of LLMs by representing weights and activations with lower-precision numerical formats instead of standard 16-bit or 32-bit floats. By compressing the model into fewer bits, quantization enables faster inference and more efficient deployment on GPUs, CPUs, and edge hardware.

In this work, quantization is applied using the BitsAndBytes NF4 (4-bit NormalFloat) format, which preserves the statistical distribution of weights while significantly reducing storage requirements. NF4 is specifically designed for neural network weights that follow a normal distribution, providing information-theoretically optimal bin placement for minimal quantization error [25]. This approach allows LLMs to maintain strong performance even under aggressive compression, making them more accessible for real-world, resource-limited environments. This approach builds upon earlier work in low-bit quantization [41] and has been shown to maintain model quality while achieving significant compression [25].

Inference-Only Constraint: BitsAndBytes 4-bit quantization produces inference-only models [25]. The quantized weights are stored in a packed format that does not support gradient computation, similar to other post-training quantization methods [42]. This constraint is critical for determining valid compression orderings, as any technique requiring training (KD or post-pruning fine-tuning) must occur before final quantization.

2.2. Compression Ordering Strategies

The baseline single technique models viz., KD [27], P [37], Q [25] show how these techniques individually help lower model complexity or computational load while still retaining reasonable accuracy yet they have their own drawbacks when used alone. These techniques suffer performance loss from heavy quantization, limited expressiveness in distilled models, or architectural imbalance due to excessive pruning. To overcome these issues, we present a unified three-step framework that combines the advantages of all three methods in a cohesive pipeline. We systematically evaluated different orderings of the three compression techniques viz., KD, P, Q to identify optimal strategies, addressing a gap in the literature where most works study these techniques in isolation [28, 29]. These distinct compression sequences were implemented and compared:

2.2.1. Three-Technique Sequences (Primary Focus)

Direct Sequences (no intermediate quantization):

- KD-P-Q: Knowledge Distillation → Pruning → Quantization
- P-KD-Q: Pruning → Knowledge Distillation → Quantization

Dequantization-Based Sequences (quantization in middle or beginning):

- Q-P-KD: Quantization → Dequantization → Pruning → Knowledge Distillation → Re-quantization
- Q-KD-P: Quantization → Dequantization → Knowledge Distillation → Pruning → Re-quantization
- KD-Q-P: Knowledge Distillation → Quantization → Dequantization → Pruning → Re-quantization
- P-Q-KD: Pruning → Quantization → Dequantization → Knowledge Distillation → Re-quantization

Please note that the dequantization-based sequences involve an implicit dequantization step (D) that is not shown in the sequence notation for brevity. For example, Q-P-KD actually represents Q-D-P-KD-Q.

2.3. Rationale for Compression Ordering

The selection of compression orderings was guided by the following theoretical considerations:

1. Quantization Last Principle: BitsAndBytes 4-bit quantization produces inference only models [25], making it incompatible with subsequent training based techniques. Therefore, all valid sequences must have **Q** as the final step, except for dequantization-based approaches. This constraint is common to most post-training quantization methods [42].
2. Dequantization Quality Concerns: Dequantization introduces information loss and potential weight corruption due to the irreversible nature of quantization [43]. Sequences with early quantization (Q-*) test the hypothesis that this quality degradation significantly impacts subsequent compression effectiveness.

3. Knowledge Transfer Order: KD-P-Q tests whether early knowledge distillation provides a stronger foundation for pruning, while P-KD-Q tests whether pruning first creates a more efficient student for distillation. This addresses questions about the interaction between distillation and structural compression raised by recent work ([33, 26]).
4. Compression Synergy: Different orderings explore whether certain compression techniques enhance or interfere with subsequent techniques (e.g., does KD make pruning more effective by creating more robust representations?). Understanding these interactions is identified as a key challenge in the compression literature [28, 44].

These ordering strategies allow comprehensive analysis of compression technique interactions and identification of optimal workflows for practical deployment scenarios, addressing a gap noted in recent surveys [29].

3. Results and Analysis

3.1. Experimental Setup

3.1.1. Hardware and Software Configuration

All experiments were conducted on a single NVIDIA A6000 GPU with 48 GB CUDA memory. The base model used for compression was Qwen2.5-3B [31], a state-of-the-art large language model with 3 billion parameters. The experimental framework was implemented using PyTorch 2.0+ [45] with the Transformers library v4.36+ [36] and BitsAndBytes v0.41+ [25] for quantization operations. Training precision utilized BFloat16 (BF16) when supported by hardware, following the mixed precision training paradigm [46], otherwise defaulting to FP32 for numerical stability.

3.1.2. Dataset

We utilized the HuggingFaceH4/ultrachat_200k dataset [47] for all training, calibration, and some evaluation procedures. The dataset consists of multi-turn conversational data suitable for chat-based language model fine-tuning. For knowledge distillation experiments, 10% of the training split (approximately 10000 samples) was used, following standard practices in neural network distillation [33]. For structured pruning experiments, 2000 samples were allocated, with 500 samples dedicated to importance score calculation and 1500 samples for post-pruning fine-tuning, consistent with calibration requirements in recent pruning literature. All text sequences were

tokenized with a maximum length of 512 tokens and padded to maintain uniform batch processing. For evaluation, we used the Stanford Question Answering Dataset (SQuAD), a reading comprehension dataset consisting of questions posed on Wikipedia articles [48]. The dataset includes a. context selection selected: Full paragraphs from Wikipedia articles, b. Question formulation: Questions created by crowdworkers based on the paragraphs, and c. Answer extraction: Spans of text that answer the questions, annotated by humans.

3.2. Evaluation Metrics

The performance of the base LLM model and its compressed version using the single baseline compression techniques as well as the three-techniques sequence is evaluated using three state-of-the-art, and widely accepted metrics. All models were assessed on perplexity, model size (MB), and three quality metrics: G-Eval score, prompt alignment, perplexity and clarity. G-Eval is a framework that leverages an LLM-as-a-judge approach with chain-of-thought reasoning to assess model outputs using any user-defined criteria [49]. In this study, correctness and relevance were adopted as core evaluation criteria to assess the factual accuracy and contextual alignment of candidate LLM-generated responses. Google’s Gemini model is used as the judge model. Perplexity (PPL) is a common metric used to evaluate language models, and it measures how well a model predicts a sequence of words. A low perplexity score means the model is less confused. The Clarity metric is a composite of three sub-metrics that together viz., fluency, contextual coherence, and readability which quantify the linguistic and contextual quality of the LLM’s response. Fluency is measured using DistilGPT2 to calculate the perplexity of the response as shown in the eq. 3. The range of the fluency score is between 0 and 1, where higher values indicate greater fluency. Contextual coherence evaluates the cosine similarity between the two vector embeddings of the LLM’s response and context using the sentence transformer. Readability is computed using the Flesch Reading Ease (FRE) score. It is a rule-based metric that evaluates sentence length and syllable count to quantify the ease of reading a text. Finally, the clarity is computed as the average of Fluency, Contextual Coherence, and Readability, ensuring equal weight across the sub-dimensions. The prompt alignment metric employs an LLM-as-a-judge approach to assess how well your LLM’s generated outputs adhere to

Table 1: Comprehensive Evaluation Results

Model	G-Eval	Prompt Alignment	Clarity	Size (MB)	Perplexity	Compression Ratio
Base Model	0.830	0.670	0.537	5886.01	3.418	1.0×
KD	0.790	0.800	0.535	5886.01	3.226	1.0×
P	0.650	0.330	0.499	4492.98	5.086	1.31×
Q	0.540	0.560	0.515	1959.44	3.955	3.00×
P-KD-Q	0.733	0.300	0.492	1600.13	5.048	3.68×
KD-P-Q	0.644	0.480	0.504	1600.13	5.553	3.68×
P-Q-KD	0.610	0.366	0.491	1600.13	5.597	3.68×
KD-Q-P	0.146	0.133	0.520	1600.13	53.366	3.68×
Q-KD-P	0.080	0.000	0.503	1600.13	24.069	3.68×
Q-P-KD	0.060	0.000	0.501	1600.13	34.494	3.68×

the instructions defined in the prompt template.

$$\text{Fluency Score} = 1 - \frac{1}{1 + \log_2 (\text{PPL})} \quad (3)$$

3.3. Overview

Table 1 presents the comprehensive evaluation results across all compression strategies. We evaluated ten model configurations: the base Qwen2.5-7B model, three single-technique baselines (KD, P, Q), and six three-technique orderings. KD’s ability to improve prompt alignment beyond the base model suggests that the distillation process with Qwen2.5-7B as teacher successfully transferred instruction-following capabilities, consistent with findings in DistilBERT. Quantization’s superior size reduction with minimal perplexity degradation demonstrates its dominant compression technique in practice. All six three-technique sequences achieved identical compression ratios ($3.68\times$, 1600.13 MB), as expected since they apply the same operations. However, performance varied dramatically across orderings, revealing critical ordering dependencies. Among the six three-technique sequences, the sequence P-KD-Q showed exemplary performance compared to others. This applies pruning first, recovers performance through knowledge distillation, and finalizes with quantization. The other sequences with early quantization sequence i.e., Q-P-KD, Q-KD-P, and KD-Q-P exhibited catastrophic performance collapse, with perplexities 5 – 15× higher than successful sequences. These failures stem from dequantization quality degradation. When models are quantized early, they must be dequantized to enable subsequent training

operations. Following are the key insights analyzed from the results shown in table 1:

- Quantization dominates single techniques: Q provides $3\times$ compression with minimal quality loss (G-Eval: 0.540), far superior to pruning’s $1.31\times$ compression at 0.650 G-Eval.
- Three-technique sequences enable aggressive compression: P-KD-Q achieves $3.68\times$ compression while maintaining 0.733 G-Eval, outperforming all single techniques except KD (which provides no compression).
- Diminishing returns: Moving from $3\times$ (Q alone) to $3.68\times$ (P-KD-Q) provides only $0.68\times$ additional compression while adding significant complexity, suggesting limited practical benefit of adding pruning to quantization-based workflows.

4. Conclusion

This study presents a comprehensive analysis of how knowledge distillation, structured pruning, and quantization interact when applied sequentially to compress Qwen LLMs. Our results show that while each technique individually contributes to model efficiency, their performance is heavily dependent on the ordering. Quantization consistently offered the highest compression with acceptable quality loss, whereas pruning introduced structural sparsity at the expense of increased perplexity. Among all evaluated sequences, P-KD-Q emerged as the most effective, achieving substantial compression while maintaining high G-Eval and clarity scores. Sequences featuring early quantization exhibited severe degradation due to irreversible quantization noise affecting downstream training. These findings highlight the importance of ordering-aware design when combining compression strategies. Overall, the work provides practitioners with a reliable pipeline for optimizing LLMs under constrained computational budgets.

Future research can explore adaptive pruning ratios and mixed-precision quantization, and alternate quantization techniques to further enhance compression without compromising accuracy. Additionally, extending this analysis to multimodal LLMs and larger-scale Qwen variants may reveal broader generalization patterns across architectures.

References

- [1] M. Y. Shaheen, Applications of artificial intelligence (ai) in healthcare: A review, ScienceOpen Preprints (2021).
- [2] L. Cao, Ai in finance: challenges, techniques, and opportunities, ACM Computing Surveys (CSUR) 55 (3) (2022) 1–38.
- [3] S. J. Plathottam, A. Rzonca, R. Lakhnori, C. O. Iloeje, A review of artificial intelligence applications in manufacturing operations, Journal of Advanced Manufacturing and Processing 5 (3) (2023) e10159.
- [4] K. Chowdhary, Natural language processing, Fundamentals of artificial intelligence (2020) 603–649.
- [5] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, Y. Wu, Exploring the limits of language modeling, arXiv preprint arXiv:1602.02410 (2016).
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
- [7] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training (2018).
- [8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (8) (2019) 9.
- [9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.
- [10] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers), 2019, pp. 4171–4186.

- [11] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *Journal of machine learning research* 21 (140) (2020) 1–67.
- [12] H. Touvron, T. Lavigra, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., Llama: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971 (2023).
- [13] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al., Constitutional ai: Harmlessness from ai feedback, arXiv preprint arXiv:2212.08073 (2022).
- [14] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, *Journal of Machine Learning Research* 24 (240) (2023) 1–113.
- [15] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al., Qwen technical report, arXiv preprint arXiv:2309.16609 (2023).
- [16] M. Sun, Z. Liu, A. Bair, J. Z. Kolter, A simple and effective pruning approach for large language models, arXiv preprint arXiv:2306.11695 (2023).
- [17] Y. Gu, L. Dong, F. Wei, M. Huang, Minilm: Knowledge distillation of large language models, arXiv preprint arXiv:2306.08543 (2023).
- [18] J. Lang, Z. Guo, S. Huang, A comprehensive study on quantization techniques for large language models, in: 2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC), IEEE, 2024, pp. 224–231.
- [19] Z. Yao, C. Li, X. Wu, S. Youn, Y. He, A comprehensive study on post-training quantization for large language models, arXiv preprint arXiv:2303.08302 (2023).
- [20] M. Chen, W. Shao, P. Xu, J. Wang, P. Gao, K. Zhang, P. Luo, Efficientqat: Efficient quantization-aware training for large language models, in:

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 10081–10100.

- [21] Y. Li, Y. Yu, C. Liang, P. He, N. Karampatziakis, W. Chen, T. Zhao, Loftq: Lora-fine-tuning-aware quantization for large language models, arXiv preprint arXiv:2310.08659 (2023).
- [22] Z. Wang, J. Wohlwend, T. Lei, Structured pruning of large language models, in: Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp), 2020, pp. 6151–6162.
- [23] G. Ling, Z. Wang, Q. Liu, Slimgpt: Layer-wise structured pruning for large language models, Advances in Neural Information Processing Systems 37 (2024) 107112–107137.
- [24] Y. An, X. Zhao, T. Yu, M. Tang, J. Wang, Fluctuation-based adaptive structured pruning for large language models, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 10865–10873.
- [25] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized llms, arXiv preprint arXiv:2305.14314 (2023).
- [26] X. Ma, G. Fang, X. Wang, Llm-pruner: On the structural pruning of large language models, Advances in neural information processing systems 36 (2023) 21702–21720.
- [27] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531 (2015).
- [28] X. Zhu, J. Li, Y. Liu, C. Ma, W. Wang, A survey on model compression for large language models, Transactions of the Association for Computational Linguistics 12 (2024) 1556–1577.
- [29] Z. Wan, X. Wang, C. Liu, S. Alam, Y. Zheng, J. Liu, Z. Qu, S. Yan, Y. Zhu, Q. Zhang, et al., Efficient large language models: A survey, arXiv preprint arXiv:2312.03863 (2023).
- [30] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, W. J. Dally, Eie: Efficient inference engine on compressed deep neural network, ACM SIGARCH Computer Architecture News 44 (3) (2016) 243–254.

- [31] Q. Team, et al., Qwen2 technical report, arXiv preprint arXiv:2407.10671 2 (3) (2024).
- [32] T. Dettmers, M. Lewis, Y. Belkada, L. Zettlemoyer, Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale, Advances in neural information processing systems 35 (2022) 30318–30332.
- [33] V. Sanh, L. Debut, J. Chaumond, T. Wolf, Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108 (2019).
- [34] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, Q. Liu, Tinybert: Distilling bert for natural language understanding, in: Findings of the association for computational linguistics: EMNLP 2020, 2020, pp. 4163–4174.
- [35] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, On calibration of modern neural networks, in: International conference on machine learning, PMLR, 2017, pp. 1321–1330.
- [36] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations, 2020, pp. 38–45.
- [37] D. Filters’Importance, Pruning filters for efficient convnets, arXiv preprint arXiv:1608.08710 (2016).
- [38] W. Kwon, S. Kim, M. W. Mahoney, J. Hassoun, K. Keutzer, A. Gholami, A fast post-training pruning framework for transformers, Advances in Neural Information Processing Systems 35 (2022) 24101–24116.
- [39] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, arXiv preprint arXiv:1611.06440 (2016).
- [40] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Advances in neural information processing systems 28 (2015).

- [41] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 2704–2713.
- [42] E. Frantar, S. Ashkboos, T. Hoefer, D. Alistarh, Gptq: Accurate post-training quantization for generative pre-trained transformers, arXiv preprint arXiv:2210.17323 (2022).
- [43] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, T. Blankevoort, A white paper on neural network quantization, arXiv preprint arXiv:2106.08295 (2021).
- [44] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, H. Jin, T. Chen, Z. Jia, Towards efficient generative large language model serving: A survey from algorithms to systems, ACM Computing Surveys 58 (1) (2025) 1–37.
- [45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Advances in neural information processing systems 32 (2019).
- [46] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, et al., Mixed precision training, arXiv preprint arXiv:1710.03740 (2017).
- [47] N. Ding, Y. Chen, B. Xu, Y. Qin, Z. Zheng, S. Hu, Z. Liu, M. Sun, B. Zhou, Enhancing chat language models by scaling high-quality instructional conversations (2023). arXiv:2305.14233.
- [48] P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang, SQuAD: 100,000+ questions for machine comprehension of text, in: J. Su, K. Duh, X. Carreras (Eds.), Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, 2016, pp. 2383–2392. arXiv:1606.05250, doi:10.18653/v1/D16-1264.
URL <https://aclanthology.org/D16-1264>
- [49] J. Ip, K. Vongthongsri, deepeval (Oct. 2025).
URL <https://github.com/confident-ai/deepeval>